Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 5 (2011) 570–577

Procedia Computer Science

The 8th International Conference on Mobile Web Information Systems (MobiWIS)

Towards Supporting Users in Semantic Exploration of Large Distributed Schemas

Mandana Sotoodeh[a,*], Rachel Pottinger[b], Philippe Kruchten[a]

[a]*Department of Electrical and Computer Engineering, University of British Columbia, 2332 Main Mall, Vancouver V6T 1Z4, Canada*
[b]*Department of Computer Science, University of British Columbia, 2366 Main Mall, Vancouver V6T 1Z4, Canada*

Abstract

Emergency management increasingly depends on various information systems to support crisis operations and facilitate communication and coordination. In this paper, we address the issues that may arise in large schemas that are developed collaboratively by diverse community of users for data sharing. We share our experience with the U.S. National Information Exchange Model and the free web-based tools available for searching its schemas. Based on what we learned, we propose techniques for enhancing the tools designed for search and exploration of such complex schemas.

Keywords: Emergency management; Data standards; NIEM; Schema exploration; Schema matching

1. Introduction

Emergency management increasingly depends on various information systems that support actors in diverse tasks such as humanitarian aid management, volunteer management, victim information management, damage assessment, geographical data visualization, and emergency warnings dissemination. Communication and coordination is at the core of emergency operations, and real-time flow of critical information is vital for an effective response. Interoperability among information systems in emergency management is mainly achieved by standardizing the format of messages or the data items composing the content of messages.

Many standards are used in emergency management, including both those tailored to other domains and those developed specifically for emergency management. These standards are typically developed and adopted before emergencies in order to set up exchange messages and to align information exchange needs in emergency situations. For example, OASIS Emergency Data Exchange Language (EDXL) [1] is a family of XML standards that defines a format for transporting and routing critical emergency messages. OASIS Customer Information Quality (CIQ) [2] defines a set of XML specifications for parties (person/organization) and their relationships. It is also used within the EDXL set of standards. These standards are complex as they contain over one hundred concepts that are shared

* Corresponding author. Tel.: +1-604-516-9193.
E-mail address: mandanas@ece.ubc.ca.

Selection and/or peer-review under responsibility of Prof. Elhadi Shakshuki and Prof. Muhammad Younas.
doi:10.1016/j.procs.2011.07.074

among messages that connect actors in different domains with little tool support to facilitate understanding of them. National Information Exchange Model (NIEM) [5], initiated by the U.S. government, is the most comprehensive emergency management data specification. Its focus is on specification of data, and its goal is enabling data sharing across all levels of the U.S. government in day to day operations as well as in the crisis. NIEM is large, containing over 6000 data components [6], specifying a wide range of concepts such as arrest, vehicle, student, commodity, etc., which can be configured in different exchange messages.

While these standards are gradually being adopted and maturing, the data sharing that motivated the creation of these standards is still missing. The standards are developed by different organizations with different viewpoints and result in inconsistent or overlapping specifications. For example, OASIS and NIEM both provide specifications for *person*, *organization*, and *address*; however, NIEM follows ISO 11179 standard for metadata definition [7], while OASIS does not, and this leads to different specifications for those concepts. Moreover, there are still gaps between the needs of the information systems for exchange of critical data and the existing standards [8].

Our goal is to support users in exploring large schemas that are distributed across several schemas representing different domains with common concepts. These schemas are to be used for sharing data and meaning, and therefore, they are designed in the way to be understood and used by humans as well as machines. Different domains may have different perspectives on shared concepts, so the users have to deal with common concepts that are modeled differently in different domains, or common concepts that are re-stated in multiple domains. Users need to understand the schema, navigate through it, and find proper data items by investigation their meaning and purpose. A proper tool support promotes adoption of these complex standards and brings about the benefits for which these schemas are initially designed. In order to gain some insights into the type of problems that may arise in specifications of such schemas and the type of challenges that users may experience, we decided to thoroughly investigate NIEM as a case study. NIEM is an umbrella for a set of XML schemas that are developed collaboratively by communities from several domains such as justice, immigration, emergency management, and others. Although it has a "Name and Design Rule" document that defines conformance to NIEM and facilitates coherency among its schemas, the flexibility it offers permits similar data items creep into the schemas. An example is the nationality for a person is represented by two different data elements: (1) *PersonNationalityText*, which is a text type (2) *PersonNationality*, which is an abstract type substitutable by data elements of different types. We argue that the tools should recognize the presence of such similarities and support users to overcome the challenges they cause.

In the next section, we look closely into NIEM and the free tools provided for searching and exploring its schemas. We report our experience with these tools and provide examples of the similarity cases we found in NIEM. In Section 3, we outline the patterns of similarities we compiled from this case study, and discuss the features that search and exploration tools should possess in order to better support users in working with similar schemas with similarity cases. We argue that these similarity patterns can happen in any schemas that adopt a similar development method as NEIM. Sections 2 and 3 address the main contributions of this paper: a characterization of problems found in understanding NIEM (Section 2); these problems are common across many large schemas and a proposed method for addressing the problems found (Section 3). Section 4 outlines our methodology to implement the tool's features described Section 3. Section 5 provides summary and future research directions.

2. Motivational example – NIEM

2.1. NIEM Schema exploration tools

In this section, we discuss the free web-based search and exploration tools available for NIEM Version 2.1. We studied these tools in order to learn what features they offer, from the search and match point of view, and where they fail to provide the maximum potential of NIEM to users.

The tools we discuss here are NIEM Wayfarer 2.1 [9], which also has a standalone version named NIEM Saw [10], NIEM SSGT [11], and the XML search tool for NIEM 2.1 on Schema Central (we refer to it as NIEM Schema Central) [12]. There are some graphical tools [13, 14] too; however, textual search tools have advanced search and exploration features and are more common to use. NIEM also captures its data dictionary neatly in a spreadsheet [15] (we refer to it as NIEM spreadsheet), which is used as a complement to the above tools for exploratory browsing. The tools are similar in basic functions but different in details. In following, we discuss their general

features and their main discrepancies. Our focus is on the features primarily used for search and exploration and not for creating an exchange package or obtaining detailed XML-specific aspects. We use *namespace:DataItem* notation to refer to a data item within a NIEM namespace. Since NIEM follows the ISO 11179 standard, a data item can be an object or a property that defines an object. We use 'type' and 'element' to refer to objects and properties. If the name of a data item ends to Type, it is a type, otherwise it is an element.

The tools start by taking search terms on the search page and lexically searching through various fields such as name, definition, or enumerations (code values) across all NIEM schemas. As we mentioned before, NIEM consists of a number of schemas distributed across several namespaces. Some tools have advanced search configuration options for specifying string patterns. Some also provide contextual search for finding elements that are inherited from the parent types.

The tools display the data items found in the schemas on the search page in alphabetic order with some variations (see Fig 1.). Some tools categorize the search results into elements, complex types, simple types, and enumerations. Some others require users to be specific in which category they want to search, and return the results based on that.

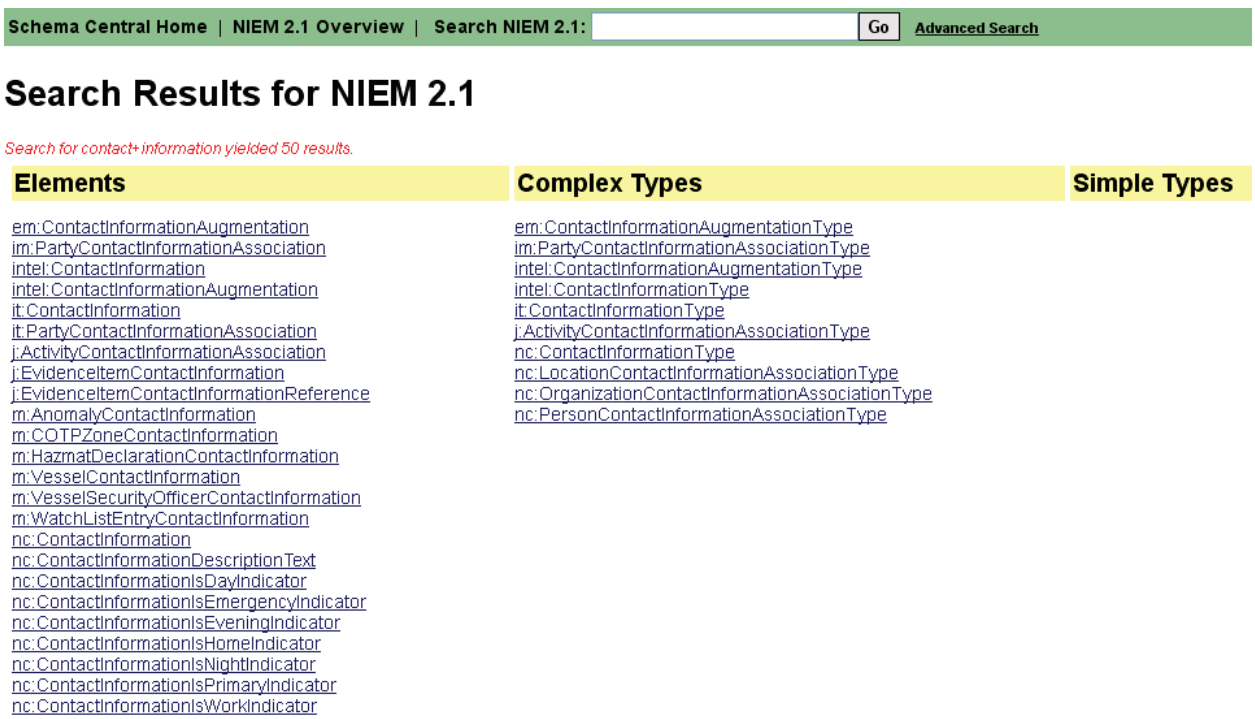

Fig. 1. Part of the search page on NIEM Schema Central: it depicts the results of search for "contact information" categorized to Elements, Complex types, Simples types, Attributes, Groups, and Attribute Groups. The last three categories were empty and excluded from this figure.

There are some other categories that are not important in this discussion. Detailed information about each data item on the search page is provided on a detailed page on the same browser.

The detailed page provides two categories of information for each data item: (1) local context that includes name, definition, local properties (for complex types), code values (for simple types), and some other details that we leave out in this discussion, (2) hierarchal context that specifies the relationships between the given data item and other data items in the schemas and its re-use information, and this is where main inconsistencies start to appear. There are mainly two types of hierarchies: type inheritance hierarchy and item containment hierarchy. A containment hierarchy represents the structure of a data item. The following points out some of these inconsistencies in displaying the hierarchal context for the complex types and elements:

- To show the inheritance hierarchy to which a type belongs, NIEM Schema Central shows the entire tree from the root to the leaves. NIEM SSGT and NIEM Wayfarer show from the root to direct children of the type and not all the way to the leaves.
- To show where in the schemas a type is used, NIEM SSGT displays the elements that are of the given type, including references to the given type. NIEM Wayfarer displays the elements that are of the given type and the derived types of the given type, excluding references to the type. NIEM Schema Central is similar to NIEM Wayfarer except that it includes references to the given type too.
- To show where in the schemas an element is used, they display the types or the elements in which the given element is contained. NIEM Schema Central displays the types in which the given element is directly used and the derived types of those types. NIEM SSGT displays the types in which the given element is directly used or referenced. NIEM Wayfarer is different from the other two tools. It displays the types in which the given element is directly used, and the types containing those types. It also displays the elements that are of the types that directly use the given element or of the derived types of those types.

One problem with the hierarchal context that is provided by these tools is that they are sometimes hard to follow. The data items are not displayed in hierarchies to which they belong, and it is not clear how a given type or element is related to other items on the screen. For example, on the detailed page for *nc:Person*, NIEM Wayfarer returns the

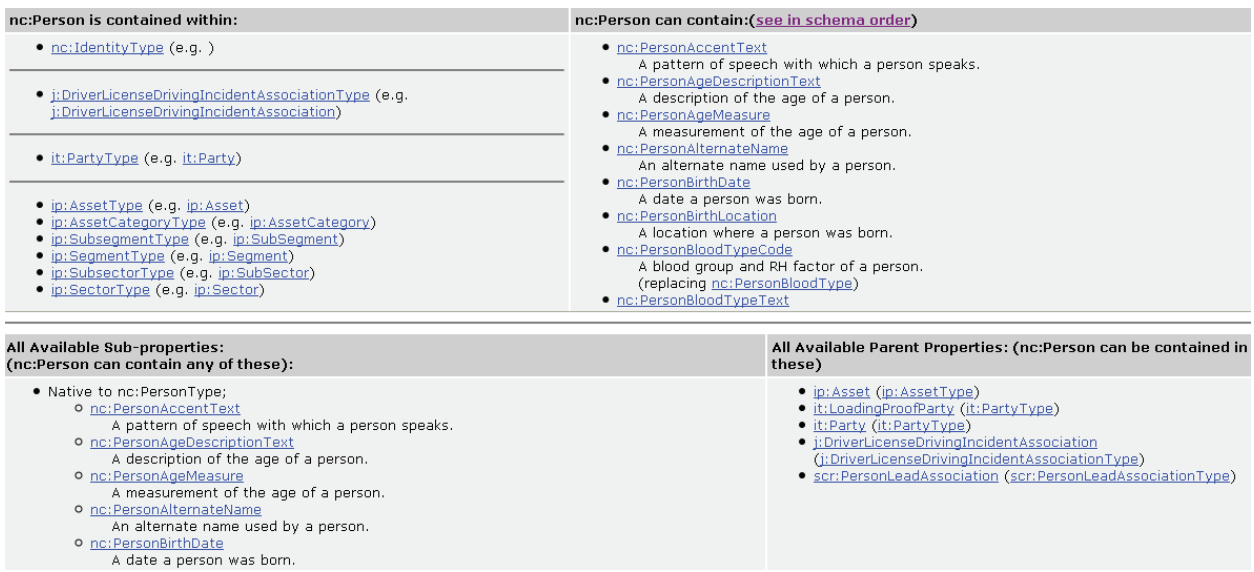

Fig. 2. Part of the detailed page on NIEM Wayfarer: it depicts the details about the nc:Person data element

following group of data items, ordered in a column, to show the types including *nc:Person*: *ip:AssetType*, *ip:AssetCategoryType*, *ip:SubsegmentType*, *ip:SegmentType*, *ip:SubsectorType*, *ip:SectorType* (see Fig 2.). It basically says that each type is contained within the next type on the list, but it is very difficult to visualize how they are related by just looking at the list. Users have to find out manually by going back and forth between pages, or looking into the NIEM spreadsheet. On the other hand, in order to understand a data item, the local information needs to be accessed first. The frequency of access to the local context and the hierarchal context may be different, and users need a clear separation between the two groups of information for ease of use. Currently this set of information is mixed together on these tools.

Another problem is that when these tools display the search results on the search page, they order them alphabetically, while there may be other relationships among them that do not follow this ordering. For example, in search for "contact information", the tools returns over 40 items including *nc:ContactInformationIsPrimaryIndicator*, *nc:ContactInformationIsEmergencyIndicator*, *nc:ContactInformationIsDayIndicator*, *nc:ContactInformationIsEveningIndicator*, and *nc:ContactInformationIsNightIndicator* (See Fig 1.). However all these elements are siblings, used inside

nc:OrganizationContactInformationAssociationType - it is almost impossible for a user to dig through all of these results and make sense of them. Categorizing the data items on the search page according to their original design facilitates sorting through a long list of items.

Finally, the tools did not explicitly collect and sort the similar data items together in order to draw users' attention and enable close comparison. If the items are sorted alphabetically irrespective of the namespace they are in, the exact equal items are listed next to each other. However if the data items mean the same but are not exactly equal, the tools do not facilitate finding them. This is the subject of the next sections.

2.2. NIEM schemas – Data item similarities

We also used the NIEM tools discussed in Section 2.1, to manually match a number of source schemas, which were available to us, to NIEM 2.1. Our goal was to find out how users use the NIEM tools to search through a set of unfamiliar schemas, start learning about them, choose a proper match for the data items in their source schema, and finally to investigate the kinds of problems the users face in this process. Although NIEM is large, but it does not necessarily mean that it addresses all the data items in a source schema. According to its documentations [16], "NIEM is a reference model. It is not a rigid standard that must be used exactly as it is in its entirety". NIEM allows users to extend its schemas, if necessary, although it may not provide clear instructions on how the extension should be done. There is always a tension between semantic clarity and size (and complexity) of schemas: more semantic distinction demands more detailed schemas. NIEM leaves the decision between choosing available data items and adding new ones to the discretion of users. However, for maximum interoperability, it is crucial users ensure to search NIEM thoroughly enough before deciding to extend it. We found two factors that interfere with this goal: (1) search efficiency and visualization effectiveness of the tools, which partly discussed in Section 2.1, (2) presence of similar data items across the NIEM schemas. Similarities cause users to have multiple options to choose from and only in a close investigation, they can choose a proper option, if there is any. In following, we discuss some of these cases in NIEM Version 2.1.

As an example, a search for "citizenship" returns over 20 data items on NIEM Wayfarer 2.1. If users do not thoroughly search the list, they may find *scr:CitizenshipBeginDate* and *scr:CitizenshipEndDate* at the top of the list and assume that NIEM does not have an element for a simple date for citizenship, while further down the list, it contains *im:CitizenshipNaturalizationDate*, an element that is defined in a different namespace.

Another example is searching for "contact person" returns over 30 data items on NIEM Wayfarer 2.1, among which is *em:ContactName* indicating "the name of the contact person or title of the contact organization". It is defined within *em:ContactInformationAugmentationType*, which is an augmentation of *nc:ContactInformationType*. *nc:ContactInformationType* is "a data type for how to contact a person or an organization", and contains an element of its own named *nc:ContactEntity*. This data element does not show on the search result on NIEM Wayfarer 2.1. However, if it is expanded, it includes *nc:PersonNameText* and *nc:OrganizationName* elements, another two possibilities for a name for a person or an organization. If the tools have the ability to highlight such a case, users can make an informed decision whether to use any of them, or ultimately extend the schema by adding elements that are more meaningful to their own context.

Another scenario is when users need to pick the data items they require for an object (or type). Consider *m:PersonAugmentationType*, which is an augmentation of a person in the Maritime domain. It contains a few data elements among which are *m:MerchantMarinerDocument* and *m:SeamanLicense*. The former specifies a merchant document for a person and the latter specifies a seaman license for a person. They both also share a number of elements from *nc:PersonType*. On the other hand, *m:PersonAugmentationType* itself is an augmentation of *nc:PersonType*, and it is already associated to all the elements in *nc:PersonType*. By highlighting this fact for users, they may decide to ignore all the person elements within *m:MerchantMarinerDocument* and *m:SeamanLicense*, and choose the ones that are directly defined within *nc:PersonType*.

3. Similarity patterns in large distributed schemas

Based on our case study from NIEM, we found some patterns of similarities in the schemas. We expect that they can exist in any large schema that is distributed across several XML schemas where each schema represents a

domain of interest. We assume that domains have overlapping concepts and schemas are structured for maximum reusability. Here we refer to each domain by a unique namespace. The similarity patters are as following:

- Duplicate types or elements: Elements are duplicates when their name and type are the same. Types are duplicates when their name and all their properties are the same. Element duplicates are more common than type duplicates. For example in NIEM, *DocumentIssueDate* is repeated in both International Trade and Screening namespaces. Users may choose the one that has more relevance to their context, or they just arbitrarily choose any of them. Here we compare the data items irrespective of the namespace in which they are defined, as data items are uniquely identifiable by their namespace.
- Type or element reuse: Reusing data elements in different containment hierarchies causes they find multiple context, and users have to consider the context when they choose their data items. For example, *nc:PersonName* element is reused in types such as *nc:PersonType*, *nc:PassportType*, and *intel:PersonInIDType*. If the type that the user is searching for is related to travel documents, the person name that is defined within *nc:PassportType* may be a better option. Data item reuse exponentially increases the number of interconnections between data items and complicates processing of such schemas.
- Elements with similar names (syntactically or semantically), and with similar or different types: An example of this case is the "contact person" that we explained in Section 2.2 Users need to study the context of the element in order to decide which option is more applicable. However there may be cases where users have to arbitrarily choose any of them.
- Types with similar names (syntactically or semantically), and with a set of similar properties: Examples for this case are *im:AlienCitizenshipType* and *intel:PersonCitizenshipDetailsType* with similar elements. They both define citizenship-related properties for a person. Again users have to consider both types in the context they are used and decide which one is a better choice for their case.

We believe existing of such similarities requires additional support in the exploration task. We define the following list of additional support for the existing tools:

- Find overlap between (the elements of) two types: it provides another way of finding similar types by inspecting the similarity of their elements than similarity of their names. For example, it helps users find out that there is *nc:JurisdictionType* which shares a number of its elements with *nc:LocationType*.
- Find similarity among the elements within a type: it helps users eliminate redundancy in their selection. An example is *m:PersonAugmentationType* in Section 2.2 where the *person*'s properties are redundant inside it.
- Find similarity among the elements across types
  - Elements along the type inheritance hierarchy: For example, *im:MarriageAssociationAugmentationType* contains *im:MarriageDateRange* element. This type is also an extension of *nc:AssociationType*, and it inherits *nc:AssociationBeginDate* and *nc:AssociationEndDate* from *nc:AssociationType*. The association date is represented differently in the parent and child types. Depending on the situation, users may prefer to choose the direct elements defined in the child type or the indirect elements inherited from the parent type. Similarity between the association dates in this case is totally semantic.
  - Elements in semantically related types or non-related types: it helps users identify the various contexts within which the similar elements are defined. An example is *nc:PersonName* that we mentioned earlier in this section.

If the schema is made of elements with primitive types, then finding similarities will be reduced to finding similar elements across the schemas.

Such patterns do not necessarily occur because the schemas are developed collaboratively. Different domains may inherently have different perspectives to similar concepts, and consequently they may conceptualize them differently. Moreover, when the schema is designed for reusability, it legitimizes the data items to be reused for creation of new data items.

4. Tool support for similarity cases

As mentioned in the previous sections, the similarity of data items in schemas complicates the search and match process, since users have to decide between multiple options. We need tools to bring these cases to the attention of users, so they can compare their options carefully. Enhancing the existing tools with this type of support can also help the developers of these schemas pinpoint and rectify such cases, if needed. In order to do that, we need more

sophisticated techniques from data integration research than simple lexical search currently being used in the existing tools. We chose to use a data integration tool suite, OpenII [17], since (1) it is open source and aligned with the goal of emergency management that encourages public participation and adoption, (2) it employs a number of syntactic and semantic (WordNet-based) string comparison techniques, (3) it is architecturally extensible, it works around a neutral data model providing a unified view to all schemas, and it also supports additions of new components such as new matchers, (4) it provides a user interface that visualizes the structure of schemas and the relationships, and (5) it has a semi-automatic approach that allows user interaction with the system.

Our goal is not to eliminate the users from the process but to provide functionality that becomes necessary as the schemas become more complex. We plan to extend OpenII with more string comparison techniques, such as SoftTFIDF evaluated by Cohen et al. [18], as OpenII's existing string matching methods are inclined towards higher recall, and it uses similar techniques to compare the name of data items and their annotation. We will evaluate them in two different settings: (1) by comparing the schemas together, (2) by combining all schemas and clustering them [19]. Since we do not assume the availability of data values, we need to find other evidence to increase the precision. We plan to find out how we can use the schema context to increase precision, and to start, we will tailor it to NIEM. The structural matching such as similarity flooding [20] that are commonly used in data integration systems may not respond well to NIEM, as the domains are not similar, and the hierarchies may not exactly match. We need to investigate what contextual information is a good candidate in our case [21, 22]. We also need to investigate how to involve users effectively and learn from their feedback to improve the search results. OpenII currently uses user feedback, by accepting and rejecting the matches, for tuning the parameters of the matchers [23]. Finally, we plan to extend OpenII with the schema exploration functionality we enumerated in Section 3.

5. Summary and future directions

Motivated by the complexity of the existing data sharing standards for emergency management, we did a thorough case study on U.S. National Information Exchange Model (NIEM) [5] to investigate the type of problems that may arise in specifications of such standards and the type of challenges that users may experience in working with them. We shared our experience with the NIEM free web-based tools and discussed their shortcomings in effectively searching and exploring its schemas. We also discussed the similarity patterns that are dominant in NIEM and proposed the functionality that the search and exploration tools should possess in order to support users to overcome the challenges imposed by these similarities. Future work will focus on fully implementing the recommendations of this paper. Ultimately we would like to learn how to help users to choose among multiple similar options, and if what the schemas offer do not satisfy their needs, how to guide users to extend the schemas.

Acknowledgements

This research was partially funded by a research fellowship in honour of Stuart Nesbitt White from Public Safety Canada.

References

1. OASIS Emergency Management TC. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=emergency.
2. OASIS Customer Information Quality (CIQ) TC. http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=ciq.
3. Cooperative Alert Information and Resource NotificationSystem. http://cairns.sourceforge.net.
4. Pack, D. and C. Coleman. Assessing interoperability in emergency management standards. in Southeastcon, 2008. IEEE. 2008.
5. National Information Exchange Model. http://www.niem.gov.
6. Priscilla Walmsley. Creating a NIEM IEPD, Part 3: Extend NIEM. 2010; http://www.ibm.com/developerworks/xml/library/x-NIEM3/?ca=drs-.
7. ISO/IEC 11179. Metadata Registry Standard. http://en.wikipedia.org/wiki/ISO/IEC_11179.
8. W3C Emergency Information Interoperability Framework Incubator Group. http://www.w3.org/2005/Incubator/eiif/XGR-EIIF-20090806.
9. NIEM Wayfarer 2.1. http://www.ncsconline.org/niemwayfarer21.
10. NIEM Saw. http://ncsconline.org/wikis/niemsaw/index.php?title=NIEM_SAW.
11. NIEM Schema Subset Generation:Search. http://tools.niem.gov/niemtools/ssgt/index.iepd.

12. Schema Central NIEM 2.1. http://www.schemacentral.com/sc/niem21/ss.html.
13. NIEM Data Model Browser. http://tools.niem.gov/niemtools/viewer/DataModelViewer.iepd.
14. Visual NIEM. http://tomcarlsonconsulting.com/visualniem.
15. NIEM Data Dictionary. http://www.niem.gov/niem/index.html.
16. NIEM Implementation Guidelines. http://www.niem.gov/implementationguide.php.
17. Open Information Integration (OpenII). http://openii.sourceforge.net.
18. William Cohen, Pradeep Ravikumar, and Stephen Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. in IJCAI '03 Workshop on Information Integration. 2003.
19. Wensheng Wu, Clement Yu, AnHai Doan, and Weiyi Meng. An Interactive Clusteringbased Approach to Integrating Source Query Interfaces on the Deep Web. in SIGMOD '04. 2004.
20. Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm and its Application to Schema Matching. in ICDE '02. 2002.
21. Jayant Madhavan, Philip A. Bernstein, AnHai Doan, and Alon Halevy. Corpus-Based Schema Matching. in ICDE '05. 2005.
22. Hong-Hai Do and Erhard Rahm, Matching large schemas: Approaches and evaluation. Information Systems, 2007. 32(6): p. 857-885.
23. Peter Mork, Len Seligman, Arnon Rosenthal, Joel Korb, and Chris Wolf, The Harmony Integration Workbench. Journal on Data Semantics XI, 2008: p. 65-93.